The Compression of Subsegments of Images Described by Finite Automata

Juhani Karhumäki * **Wojciech Plandowski** † **Wojciech Rytter** ‡

Abstract

We investigate how the size of the compressed version of a 2-dimensional image changes when we cut off a part of it, e.g. extracting a photo of one person from a photo of a group of people. 2-dimensional compression is considered in terms of finite automata. Let n be the size of the smallest acyclic automaton which describes an image T. We show that the tight bound for the compression size of a subsegment (subimage) in the deterministic case is $\Theta(n^{2.5})$ and in the weighted case is $\Theta(n)$. We also show how to construct efficiently the compressed representation of subsegments given the compressed representation of the whole image. Two applications of subsegments compression are more efficient automata-compressed pattern-matching and the first polynomial time algorithm for the fully compressed pattern-checking problem for weighted automata.

1 Introduction

The compression size of images is of crucial importance in multimedia systems and in transferring large images in WWW. Deterministic and weighted finite automata are successful tools for compressing 2-dimensional images, see [3, 4, 5, 7]. There are several software packages using this type of compression, see [9, 4]. Finite automata can describe quite complicated images, for example deterministic automata can describe the Hilbert's curve with a given resolution, see [11], while weighted automata can describe even much more complicated curves, see also [3, 4, 5]. The objects considered are potentially exponentially compressed, so algorithms which apply decompression are theoretically not polynomial time algorithms. In practice exponential compression does not usually appear, nevertheless the compression ratio for two-dimensional images can be very high, especially compared with the one dimensional case (for example for images corresponding to fractals having short description). For one-dimensional words there exist polynomial-time deterministic algorithms

*Department of Mathematics, Turku University, Finland. Supported by Academy of Finland under grant 14047. Email:karhumak@cs.utu.fi.

†Turku Centre for Computer Science, DataCity 4th floor, Lemminkäinsenkatu 14, FIN 20 520, Finland. On leave from Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02–097 Warszawa, Poland. Email:wojtekpl@mimuw.edu.pl.

‡Instytut Informatyki, Uniwersytet Warszawski, Banacha 2, 02–097 Warszawa, Poland, and Department of Computer Science, University of Liverpool. supported partially by the grant KBN 8T11C03915. Email:rytter@mimuw.edu.pl

for compressed and fully compressed pattern-matching [8, 10, 12], despite the fact that the uncompressed size of objects could be exponential. However these problems become much harder in the two-dimensional case. Our main result is a constructive proof of the fact that compressed size of subsegments of an automata-compressed images grows only polynomially. This contrasts with the exponential grow of compression size of subimages for compression in terms of recursive description, see [2]. Our alphabet is $\Sigma = \{0,\ 1,\ 2,\ 3\}$, the elements of which correspond to four quadrants of a square array, see Figure 1.

1	3
0	2

Figure 1: Enumeration of the quadrants.

A word w of length k over Σ can be interpreted, in a natural way, as a unique address of a pixel x of a $2^k \times 2^k$ image (array), we write $address(x) = w$. The length k is called the *resolution* of the image. For a language $L \subseteq \Sigma^+$ denote by $Image_k(L)$ the $2^k \times 2^k$ black-and-white image such that the color of a given pixel x is black iff $address(x) \in L$. We consider also the weighted languages, formally they correspond to functions which associates with each word w a value $weight_L(w)$. A weighted language L over Σ and resolution k determine the gray-tone image $Image_k(L)$ such that the color of a given pixel x equals $weight_L(address(x))$. If all words in L are of the same length k then we can omit the subscript k and write $Image(L)$. Our description of the language is in terms of finite (unweighted or weighted) automaton A. We define $Image_k(A) = Image_k(L(A))$, where $L(A)$ is the language accepted by A.

Representation in terms of acyclic deterministic automata is equivalent to a representation by a 2-dimensional grammar, each production corresponds to the way of decomposing a square into 4 smaller subsquares of a same shape. For the automaton from Figure 2 we can define the subsquare corresponding to state $s1$ by ($\hat{\emptyset}$ denotes a blank subsquare of appropriate shape):

$$s_1 \to \begin{bmatrix} s_2 & \hat{\emptyset} \\ s_2 & s_2 \end{bmatrix}$$

We consider a subsegment image P and the host image T described by automata of sizes m and n. Denote by $Compress(P)$ and $Compress(T)$ the automata describing P and T, respectively. Our main problem is the **Subsegment Compression Problem**:

Instance: $Compress(T)$ representation of a $2^k \times 2^k$ square image, a point x in T and an integer $k' < k$. Let R be a square $2^{k'} \times 2^{k'}$ subsegment of T whose left-upper corner is positioned at x.

Question: What is the size of $Compress(R)$? What is the complexity of computing $Compress(R)$.

Another problem is the pattern-checking, it consists in testing a fixed occurrence of a large compressed image.) It is co-NP complete for 2-dimensional compressions in terms of recursive generations, see [2]. We define the depth of the automaton as the longest path from the initial state to an accepting state. An acyclic automaton can be transformed to an equivalent automaton in which for each state q each path from the initial state to q has the same length. We say that a state q belongs to level t if all paths from the initial state to q are of length t. Here, the length of a path is the number of edges in the path. In our considerations we may restrict to acyclic automata since we consider only finite resolution images or more precisely finite resolution approximations of infinite resolution images.

Example. $Image(\{0,1,2\}^k) = S_k$ is the $2^k \times 2^k$ black-and-white square part of Sierpinski's triangle, see Figure 2 for the case $k = 4$. The corresponding smallest acyclic deterministic automaton accepting all paths describing black pixels has 5 states.

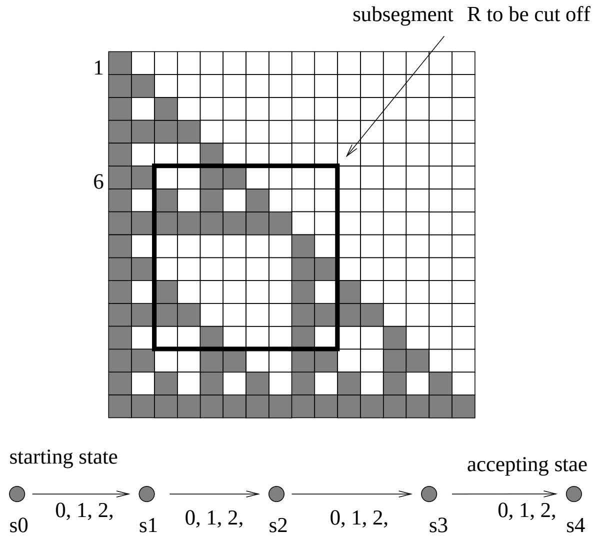

Figure 2: The image S_4 and its smallest acyclic automaton. Edges which are not on accepting paths are disregarded.

2 The Subsegment Compression Problem for Deterministic Automata

Let $A = (\{0,1,2,3\}, Q, q_0, \delta)$ be a deterministic acyclic automaton of depth n where Q is a set of states, $q_0 \in Q$ is the initial state, and $\delta : Q \times \Sigma^* \to Q$ is a partial transition function. The automaton A defines the language $L(A) = \{w : \delta(q_0, w)$ is defined $|w| = n$ and $\delta(q_0, w)$is accepting$\}$. Note, that in the definition of an automaton we do not need to specify the single accepting state. For $q \in Q$, denote by $Image(q)$ the image which is generated by the automaton which is obtained from A by changing its initial state to q. Clearly, $Image(q_0) = Image(A)$.

A regular block of a $2^k \times 2^k$ image T is defined as follows. T is a regular block, and if B is

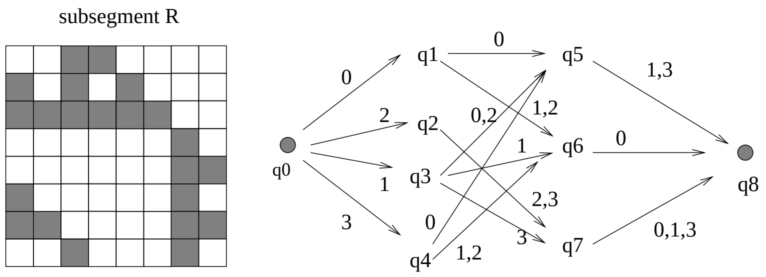

Figure 3: The subsegment R of S_4 and the smallest acyclic automaton describing R.

a regular block then all its quadrants are also regular blocks. A square subsegment of the shape $2^t \times 2^t$ is said to be of *rank* t. Denote by $\hat{\emptyset}$ a square blank block consisting only of white pixels. We use the same notation for all possible sizes of $\hat{\emptyset}$, the size depends on the context. We can interpret the state q as a name of a regular block $X = Image(q)$, we write $name(X) = q$. If X is a blank block then we write $name(X) = \hat{\emptyset}$. We call states to be essential iff they are on a path to an accepting state.

Lemma 2.1 *Assume the whole image is not totally blank. The number of essential states of the smallest acyclic deterministic minimal automaton describing T equals the cardinality of different nonblank regular blocks of T.*

We illustrate the lemma with the following example. The states at depth 2 of the automaton from Figure 3 corresponds to the blocks of the subsegment R in the way shown in Figure 4.

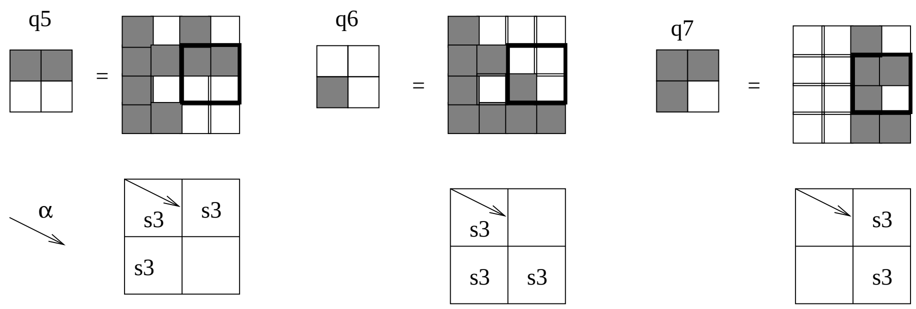

Figure 4: Illustration of Lemma 2.1. Regular blocks of R (in bold) are parts of pseudo-blocks of T. There are 3 nonblank regular blocks of rank 1 in segment R from Figure 3, they correspond to the states of the corresponding automaton for R, each of these blocks is a part of a psuedo-regular block of rank 2 of T with the same vector α shown in the figure. We have $q5 = Sub(\alpha, s3, s3, \emptyset, s3)$, $q6 = Sub(\alpha, s3, s3, s3, \emptyset)$, $q7 = Sub(\alpha, \emptyset, \emptyset, s3, s3)$.

The crucial notion is that of a *pseudo-regular block*, defined as a square subsegment of a rank $t+1$ of the corresponding image consisting of 4 adjacent *regular* blocks of rank T, the regular blocks themselves are also considered as pseudo-regular blocks. Define by $Pseudo\_Reg_t(T)$ the set of pseudo-regular blocks of rank t of T. For a subblock X of a block X define the position of X in Y ($pos_X(Y)$) as the position of left-upper corner of Y inside X. For a block X of rank $t+1$ and vector α define $Sub_t(\alpha, X)$ to be the sublock Y of

X such that $pos_X(Y) = \alpha$. Each pseudo-regular block X is identified by a composite name $(name(X_1), name(X_2), name(X_3), name(X_4))$ where $X_1, \ldots X_4$ are regular blocks which are quadrants of X, listed in the order corresponding to Figure 1. The compressed size of the subsegment can grow since the number of pseudo-regular blocks could be much larger than the number of regular blocks in the same image. For example there are 2 regular blocks of rank 1 in T in Figure 2, but 5 different pseudo-regular blocks of rank 1 (including blank ones).

Lemma 2.2
For a given rank t there is a vector α_t such that each regular block of rank t in the subimage R equals $Sub(\alpha_t, (A, B, C, D))$ where (A,B,C,D) is a pseudo-regular block of T of rank $t+1$.

Theorem 2.3 (deterministic automata)
Assume the compression is in terms of deterministic automata. The compressed representation of a square subsegment R of T can be computed in $O(|Compress(T)|^{2.5})$ time.

Proof: The states of the automaton A' for the subsegment are tuples $(\alpha_k, (A, B, C, D))$ where (A, B, C, D) are pseudo-regular blocks of T of rank $k + 1$, and α_k is a vector from Lemma 2.2. Due to Theorem 3.3 the number of pseudo-regular blocks is $O(n^{2.5})$, where $n = |Compress(T)|$.

We compute names of pseudo-regular blocks top down. However we consider only these pseudo-regular blocks which contain a nonblank block of the subsegment. If we know the names of pseudo-regular blocks of rank $t+1$ then each pseudo-regular block of rank t consists of 4 subblocks of rank t of a single pseudo-regular block of rank $t + 1$, the details will be given in the full version. □

3 Tight bounds for the compression size of subimages

We need the following technical lemmas.

Lemma 3.1 *Let k_1, k_2 …, $k_r > 0$. Then*

$$\sum_{i=2}^{r} \min(k_{i-1}^4, k_i^2 + k_{i+1}^2 + \ldots + k_r^2) \leq (k_1 + \ldots + k_r)^{2.5}.$$

Proof: We omit the technical proof (induction on r). □

Lemma 3.2 *Each pseudo-regular block of rank i is a central block of a regular block of rank k, or a central block of two adjacent (horizontally or vertically) regular blocks of rank k, where $k \geq i$.*

Theorem 3.3 *For each subimage $\mathcal{R}$ of an image $\mathcal{T}$ described by a deterministic automaton of size n there is a deterministic automaton describing $\mathcal{R}$ of size $O(n^{2.5})$.*

Proof: By the construction of the proof of Theorem 2.3 it is enough to give an upper bound for the number of all pseudo-regular subsquares of $\mathcal{T}$.

Let ps_i be the number of pseudo-regular blocks of rank i for $1 \le i \le r$ and k_i be the number of regular blocks of rank i for $0 \le i \le r$. Then due to Lemma 3.2 the number of pseudo-regular blocks can be bounded by the number of pairs of regular blocks of rank at least i, hence we have

$$ps_i \;=\; O(k_i^2 + k_{i+1}^2 + \ldots + k_r^2)$$

In the same time each pseudo-regular block of rank i is composed of 4 regular blocks of rank $i-1$, hence $ps_i \le k_{i-1}^4$. Therefore

$$ps_i = O(\min(k_{i-1}^4,\; k_i^2 + k_{i+1}^2 + \ldots + k_r^2)$$

The conclusion of the theorem is now a consequence of Lemma 3.1. □

In the proof of the lower bound we need two generations of size $O(n)$ which composed together give an object whose compression size is $\Omega(n^2)$. We use the following fact. Assume here the alphabet consists of all integeres and we have morphisms

$$h(i) \;=\; 2i\ 2i\ 2i+1\ 2i+1; \quad g(i) \;=\; 2i\ 2i+1\ 2i\ 2i+1;$$

We have

$$\begin{aligned} h^2(0) &= 0011001122332233; \\ g^2(0) &= 0101232301012323. \end{aligned}$$

Lemma 3.4
Let $u = h^k(0)$ and $w = g^k(0)$. Then all n^2 pairs $(u[i], w[i])$ are different for $1 \le i \le n^2$, where $n = 2^k$.

The structure of an image whose compression size is n and compression size of its subsegemnt is $\Omega(n^{2.5})$ is illustrated in Figure 5. Lemma 3.4 is used to generate n different subsquares on one side of the middle line and n different subsquartes on its other side in such a way that we have $\Omega(n^2)$ diferent pairs of these subsquares. We can "pump" these subsquares in such a way that we receive many different subsquares which are of size $O(2^{\sqrt{n}})$ and which are blank except small shaded subsquares. There are $\Omega(n^2)$ different (shaded) small subsquares of T, each one consiting of 4 regular subsquares touching the middle line. The distance between consecutive small subsquares is $2d$, where $d = 2^{\sqrt{(n)}}$. There are $\Omega(n^2)$ $d \times d$ regular subsquares $U_1, U_2, \ldots$ in the subsegment (touching the middle line). Each small shaded corner subsquare of U_i is different (there are $\Omega(n^2)$ of them), so there are together $\Omega(n^2 \cdot \log(|U_i|)) = \Omega(n^{2.5})$ different regular subsquares (dotted subsquares in the figure). Hence the compressed size of the subsegment should be $\Omega(n^{2.5})$.

Theorem 3.5 (lower-bound)
There is an infinite sequence of deterministic automata of square images T described by deterministic automata such that $|Compress(T)| = n$ and there is a square subsegment R of T satisfying $|Compress(R)| = \Omega(n^{2.5})$.

Proof: We use the grammars to describe the image $T = \mathcal{I}_k$ (such description is equivalent to the one in terms of automata).

$$S \to \left[\begin{array}{cc} A_0^{(2k)} & A1_0^{(2k)} \\ B^{(2k)} & B^{(2k)} \end{array} \right]$$

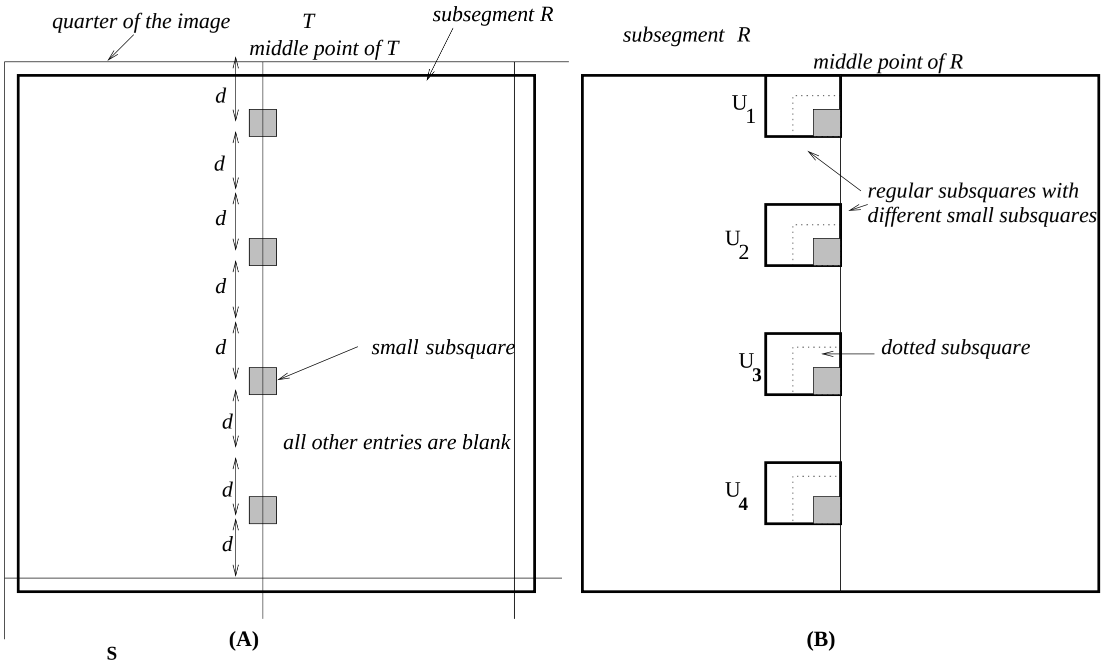

Figure 5: The structure of the image $T = \mathcal{I}_k$ and the subsegment R (indicated in bold).

$$A_i^{(2j)} \to \begin{bmatrix} B^{(2j-1)} & A_{2i}^{(2j-1)} \\ B^{(2j-1)} & A_{2i+1}^{(2j-1)} \end{bmatrix} \quad A_i^{(2j+1)} \to \begin{bmatrix} B^{(2j)} & A_i^{(2j)} \\ B^{(2j)} & A_i^{(2j)} \end{bmatrix}, \text{ for } 0 \le j < k,$$

$$A1_i^{(2j)} \to \begin{bmatrix} A1_i^{(2j-1)} & B^{(2j-1)} \\ A1_i^{(2j-1)} & B^{(2j-1)} \end{bmatrix} \quad A1_i^{(2j+1)} \to \begin{bmatrix} A1_{2i}^{(2j-1)} & B^{(2j-1)} \\ A1_{2i+1}^{(2j-1)} & B^{(2j-1)} \end{bmatrix}, \text{ for } 0 \le j < k,$$

where $B^{(j)}$ generates a square of blanks of appropriate shape. Consider a square I_k which is generated from S by applying the above productions. It consists of the symbols $A_i^{(0)}$, $A1_i^{(0)}$ and $B^{(0)}$. The following lemma holds.

Lemma 3.6
(1) *I_k is generated by $O(k)$ productions.*
(2) *The middle two-column stripe of I_k contains all pairs $A_i^{(0)}$, $A1_s^{(0)}$, for $0 \le i, s \le k-1$.*

Next part of the grammar looks as follows:

$$A_i^{(0)} \to \begin{bmatrix} B1^{(\sqrt{k})} & C_{low(i)}^{(\sqrt{k})} \\ B1^{(\sqrt{k})} & C_{high(i)}^{(\sqrt{k})} \end{bmatrix}, A1_i^{(0)} \to \begin{bmatrix} C_{low(i)}^{(\sqrt{k})} & B1^{(\sqrt{k})} \\ C_{high(i)}^{(\sqrt{k})} & B1^{(\sqrt{k})} \end{bmatrix}.$$

where $low(i)$ (lowest $k/2$ bits of i), $high(i)$ (highest $k/2$ bits of i) form a representation of the number i in the base $\sqrt{k}$. Now we apply above productions to I_k obtaining a square $I1_k$ consisting of squares $B1^{(\sqrt{k})}$, $C_i^{(\sqrt{k})}$ with the following property being a consequence of part 2 of Lemma 3.6: for $0 \le i, j, s, r < \sqrt{k}$ middle two-column stripe is composed of all possible 2×2 squares of the form

$$\begin{bmatrix} C_i^{(\sqrt{k})} & C_s^{(\sqrt{k})} \\ C_j^{(\sqrt{k})} & C_r^{(\sqrt{k})} \end{bmatrix}.$$

For each i, $0 \leq i < \sqrt{k}$, we define a unique black-white square a_i. They are of the same shape. We may generate all squares a_i using $O(\sqrt{k})$ productions. The square $C_i^{(\sqrt{k})}$ is $2^{\sqrt{k}} \times 2^{\sqrt{k}}$ square containing a_i at all its positions. To generate all of them we require $O(k)$ productions. To prove the lemma it is enough to show the following.

Lemma 3.7 *Let the sides of squares a_i be sa. Then a subimage of $\mathcal{I}_k$ which is of shape as the quarter of $\mathcal{I}_k$ and whose left upper corner is at position $(sa+1, sa+1)$ in $\mathcal{I}_k$ contains at least $k^{2.5}$ regular subsquares.*

This completes the proof of the theorem. □

4 The Subsegment Compression Problem for Weighted Automata

For weighted automata the compression size of the subsegment grows only linearly, this surprising phenomenon is due to the fact that for weighted automata we can have many edges from the same state labelled with the same symbol, but having possibly different weights. This enables to do operations similar to matrix addition, such trick is not possible in the deterministic case. A weighted finite automaton describing an image is specified by (see [9] for details): set of states Q, the alphabet $\{0,1,2,3\}$, weight of edges given by the function $W_a : Q \times Q \to (-\infty, \infty)$ for edges labeled by the symbol a, for $a \in \{0,1,2,3\}$, a function $I : Q \to (-\infty, \infty)$ called *initial distribution function* and a function $F : Q \to (-\infty, \infty)$ called *final distribution function.*

The weight of a word $w = a_1 a_2 \ldots a_k$ is interpreted to give a color $W(w)$ for the pixel $entry(w)$. It is defined as $W(w) = I W_{a_1} W_{a_2} \ldots W_{a_k} F$.

For a regular block Y of T and a vector v denote by $block(Y, v)$ the square which is identical to Y on the overlap of Y with the square of the same shape shifted from the corner of Y inside Y by the vector v, all other entries are blank (contain zeros), see Figure 6.

Theorem 4.1 (weighted automata)
Assume the compression is in terms of weighted automata. If $|Compress(T)| = n$ then for a square subsegment R of T we have $|Compress(R)| = O(n)$. The Subsegment Compression Problem for weighted automata for a square subsegment R can be computed in linear time.

Proof: (sketch)
Let A be the weighted automaton defining T, we identify its states with regular blocks. For each rank t a regular block of rank t of subsegment is located in a pseudo-regular block of T, where vectors $v1, v2, v3, v4$ are as in Figure 6.

We create the automaton A' for R. Its states are identified with $block(Y, v)$, where Y is a regular block of T and v is one of the vectors $v1, \ldots, v4$ which depend on the rank.

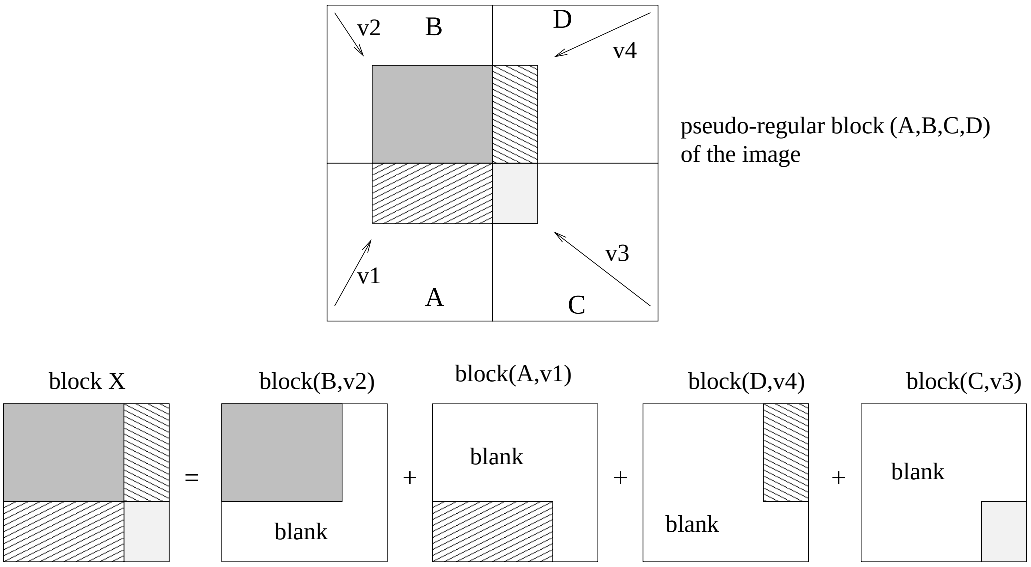

Figure 6: Regular block X cut off from a pseudo-regular block of T (as a matrix) can be treated as the sum: $X = block(A, v1) + block(B, v2) + block(C, v3) + block(D, v4)$.

Correctness is based on the fact that the subblock X can be treated as a matrix and it is the sum of 4 matrices, as shown in the figure. For each rank there are 4 different vectors, so the number of states of A' is $O(n)$. The construction goes top-down similarly as in Theorem 2.3 using the ideas from the proof of Theorem 4.2 in [11], where *atomic dependencies* are decomposed into smaller ones. In our case atomic dependencies correspond to subblocks of the type $block(Y, v)$. We omit the details. □

5 Two Applications of the Subsegment Compression

We sketch two simple consequences of the subegment compression. The first is an improvement (and simplification) upon a similar result in [11] and the second one gives the first polynomial time algorithm for compressed checking problem for weighted automata.

Theorem 5.1 *There is an algorithm for compressed pattern-matching for deterministic automata working in time $O(n^{2.5}m)$ where n is the compressed representation of T and m is the total size of an uncompressed pattern P.*

Proof: In the process of construction the representation for a subsegment we compute pseudo-regular blocks. It can be shown that P occurs in T if it occurs in a psudo-regular block of rank $t + 1$, where t is the rank of P. Hence we can construct all pseudo-regular blocks of rank $t+1$ and check for them (by known linear time algorithms for uncompressed two-dimensional matching) if P occurs in one of them. □

Theorem 5.2 *There is a polynomial time algorithm for the fully compressed checking problem for weighted automata.*

Proof:
We use the following result due to [6].

Claim. The equivalence of two weighted automata can be checked in polynomial time.

We can construct the compressed representation of the subsegment R of T which is of the same shape as the patern P and starts at the same location. Then we check equality of the images R, P in polynomial time due to the claim. □

References

[1] A. Amir and G. Benson, Efficient two dimensional compressed matching, *Proc. of the 2nd IEEE Data Compression Conference* 279-288 (1992).

[2] P. Berman, M. Karpinski, L. Larmore, W. Plandowski, W. Rytter, The compexity of pattern matching of highly compressed two-dimensional texts, Combinatorial Pattern Matching 1997, in Springer Verlag

[3] K. Culik and J. Karhumaki, Finite automata computing real functions, *SIAM J. Comp* (1994).

[4] K. Culik and J. Kari, Image compression using weighted finite automata, *Computer and Graphics* 17, 305-313 (1993).

[5] D. Derencourt, J. Karhumäki, M. Letteux and A. Terlutte, On continuous functions computed by real functions, *RAIRO Theor. Inform. Appl.* 28, 387-404 (1994).

[6] S. Eilenberg, *Automata, Languages and Machines*, Vol.A, Academic Press, New York (1974).

[7] K. Culik and J. Kari, *Fractal image compression: theory and applications*, (ed. Y. Fisher), Springer Verlag 243-258 (1995).

[8] M. Farach and M. Thorup, String matching in Lempel-Ziv compressed strings, in *STOC'95*, pp. 703-712.

[9] J. Kari, P. Franti, Arithmetic coding of weighted finite automata, *RAIRO Theor. Inform. Appl.* 28 343-360 (1994).

[10] L. Gąsieniec, M. Karpiński, W. Plandowski and W. Rytter, Efficient Algorithms for Compressed Strings, in *SWAT'96* (1996).

[11] J.Karhumaki, W.Plandowski, W. Rytter, Pattern matching for images generated by finite automata, FCT'97, in LNCS Springer Verlag 1997

[12] M. Karpinski, W. Rytter and A. Shinohara, Pattern-matching for strings with short description, in *CPM'95* (1995).

Appendix

Proof of Lemma 3.1

Induction on r. For $r = 2$ we have

$$\min(k_1^4, k_2^2) \leq k_2^2 \leq (k_1 + k_2)^{2.5}.$$

Let $r > 2$ and inductively assumme that the inequality is satisfied for $r - 1$. By induction we have

$$\min(k_2^4, k_3^2 + \ldots + k_r^2) + \ldots + \min(k_{r-1}^4, k_r^2) \leq (k_2 + \ldots + k_r)^{2.5}.$$

It is enough to prove that

$$\min(k_1^4, k_2^2 + k_3^2 + \ldots + k_r^2) + (k_2 + \ldots + k_r)^{2.5} \leq (k_1 + \ldots + k_r)^{2.5}.$$

Hence, it is enough to prove

$$\min(k_1^4, (k_2 + k_3 + \ldots + k_r)^2) + (k_2 + \ldots + k_r)^{2.5} \leq (k_1 + \ldots + k_r)^{2.5}.$$

Let $x = k_1$ and $y = k_2 + \ldots + k_r$. We have to prove that

$$\min(x^4, y^2) + y^{2.5} \leq (x + y)^{2.5}.$$

We use the following fact with $\alpha = 2.5$.

Fact 5.3 *Let x, $y > 0$ and $\alpha > 1$. Then*

$$(x + y)^{\alpha} > y^{\alpha} + \alpha y^{\alpha - 1} x.$$

Using the above formula it is enough to prove

$$\min(x^4, y^2) \leq 2.5 y^{1.5} x.$$

We consider two cases.

Case $x^4 < y^2$. Then

$$\min(x^4, y^2) = x^4 \leq x \cdot x^3 \leq x y^{1.5} \leq 2.5 y^{1.5} x.$$

Case $x^4 \geq y^2$. Then

$$\min(x^4, y^2) = y^2 \leq y^{0.5} y^{1.5} \leq x y^{1.5} \leq 2.5 y^{1.5} x.$$

This completes the proof.